

09/830588

PCT/JP00/05007

日 本 国 特 許 庁

01.08.00

PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application:

1999年 9月 9日

REC'D 12 SEP 2000

出 願 番 号
Application Number:

平成11年特許願第255272号

WIPO

PCT

出 願 人
Applicant(s):

科学技術振興事業団

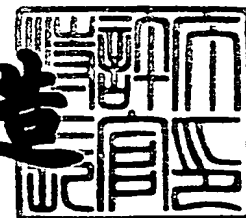
PRIORITY
DOCUMENT

SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH RULE 17.1(a) OR (b)

2000年 9月 1日

特許庁長官
Commissioner,
Patent Office

及 川 耕 造



出証番号 出証特2000-3069123

【書類名】 特許願

【整理番号】 P0159JP

【提出日】 平成11年 9月 9日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/00

【発明者】

 【住所又は居所】 千葉県千葉市稲毛区弥生町1-170-2-203

 【氏名】 松本 尚

【特許出願人】

 【識別番号】 396020800

 【氏名又は名称】 科学技術振興事業団

【代理人】

 【識別番号】 100107010

 【弁理士】

 【氏名又は名称】 橋爪 健

【手数料の表示】

 【予納台帳番号】 054885

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 アクセス方法及びアクセス処理プログラムを記録した記録媒体

【特許請求の範囲】

【請求項 1】

第 1 のプロセスが、入出力装置又はインタフェースのオープン処理をオペレーティングシステムに要求するステップと、

オペレーティングシステムが、第 1 のプロセスの要求格納領域を示すためのコンテキスト識別子を割り当て、該コンテキスト識別子に該当するメモリページを未処理の要求があることを示すペンディングレジスタのアクセス用アドレスとしてマップするステップと、

第 1 のプロセスが、要求格納領域へ入出力装置又はインタフェースへの要求内容を記述するステップと、

オペレーティングシステムが、ペンディングレジスタのアクセス用アドレスを使って、未処理の要求があることを入出力装置又はインタフェースに伝えるステップと、

入出力装置又はインタフェースが、ペンディングレジスタに記憶されたコンテキスト識別子に基づき、第 1 のプロセスの要求を読み出すステップを含むアクセス方法。

【請求項 2】

オペレーティングシステムは、入出力装置又はインタフェース内の内蔵メモリに、各プロセスに対応する要求格納領域の物理アドレスを記憶するようにしたことを特徴とする請求項 1 に記載のアクセス方法。

【請求項 3】

入出力装置又はインタフェースは、

ペンディングレジスタにアクセスがあったことを識別するステップと、

コンテキスト識別子に基づき、各プロセスの要求格納領域の物理アドレスを記憶した内蔵メモリを参照することにより、ペンディングレジスタをアクセスしたプロセスの要求格納領域の物理アドレスを得るステップと、

要求格納領域の内容を読み出して、要求内容を実現するステップと

を備えたことを特徴とする請求項 1 又は 2 に記載のアクセス方法。

【請求項 4】

入出力装置又はインタフェースを指定する物理アドレスは、ペンディングレジスタの位置を示すファンクションセレクト領域と、プロセスを示すコンテキスト識別子領域とを含み、

アドレスデコーダは、前記物理アドレスのファンクションセレクト領域にペンディングレジスタの場所を表す固有アドレスが記憶されている場合、前記物理アドレスのコンテキスト識別子をペンディングレジスタに記憶することを特徴とする請求項 1 乃至 3 のいずれかに記載のアクセス方法。

【請求項 5】

前記ペンディングレジスタに、コンテキスト識別子に対応する他のデータを必要に応じて記憶することを特徴とする請求項 1 乃至 4 のいずれかに記載のアクセス方法。

【請求項 6】

第 1 のプロセスが入出力装置又はインタフェースのクローズ処理をオペレーティングシステムに要求した場合又は第 1 のプロセスが終了した場合には、オペレーティングシステムは第 1 のプロセスに割り当てたアドレスを回収すると共に、入出力装置又はインタフェース用のコンテキスト識別子を回収し、及び又は、内蔵メモリ上の第 1 のプロセスの要求領域への物理アドレスのエントリをクリアすることを特徴とする請求項 1 又は 5 のいずれかに記載のアクセス方法。

【請求項 7】

第 1 のプロセスが、入出力装置又はインタフェースのオープン処理をオペレーティングシステムに要求するステップと、

オペレーティングシステムが、第 1 のプロセスの要求格納領域を示すためのコンテキスト識別子を割り当て、該コンテキスト識別子に該当するメモリページを未処理の要求があることを示すペンディングレジスタのアクセス用アドレスとしてマップするステップと、

第 1 のプロセスが、要求格納領域へ入出力装置又はインタフェースへの要求内容を記述するステップと、

オペレーティングシステムが、ペンディングレジスタのアクセス用アドレスを使って、未処理の要求があることを入出力装置又はインタフェースに伝えるステップと、

入出力装置又はインタフェースが、ペンディングレジスタに記憶されたコンテキスト識別子に基づき、第 1 のプロセスの要求を読み出すステップを含むアクセス処理プログラムを記録した記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、アクセス方法及びアクセス処理プログラムを記録した記録媒体に係り、特に、仮想化されたマルチコンテキスト環境におけるユーザレベル入出力アクセス方法及びアクセス処理プログラムを記録した記録媒体に関する。

【0002】

【従来の技術】

図 6 に、従来方式 1 のシステムの説明図を示す。この図では、従来のカーネルアドレス空間にマップされたシステムの説明図を示す。

【0003】

従来方式 1 では、図示のように、入出力装置（I/O 装置）（I/O device）の制御用レジスタは、汎用オペレーティングシステム（OS）の下では、カーネル（kernel）モードでしかアクセスできないようにすることにより、ユーザアプリケーションによる不正な操作を防止している。物理アドレス空間（physical address space）は、カーネルアドレス空間（kernel address space）とユーザアドレス空間（user address space）とに分かれる。I/O 装置は、カーネルアドレス空間の I/O 装置に対応するエントリがアクセスされた場合のみ、I/O 装置を使用することができる。そして、ユーザアプリケーションとしてのプロセス A が、I/O 装置を使用したい場合には、OS のシステムコール（system call）を通じて操作要求を出す。つぎに、OS が、カーネルモードにおいて操作要求の正当性をチェックした後、具体的な操作を I/O 装置内の制御レジスタに施す。

【0004】

また、最近では、ユーザアプリケーションのアドレス空間に I/O 装置の制御レジスタをマップして、ユーザアプリケーション（ユーザプロセス）が直接制御レジスタをアクセスし、入出力操作のオーバーヘッドを減らす方式も使われるようになってきている。図 7 に、従来方式 2 のシステムの説明図を示す。この図では、ユーザアドレス空間にマップされたシステムの構成図の一例を示す。この従来方式 2 では、プロセス A が、物理アドレス空間にある I/O 装置用ページ (page for I/O) を、ユーザアドレス空間にマッピングしている。I/O 装置用ページにより、I/O 装置の制御レジスタを使用することができる。

【0005】

つぎに、図 8 に、従来方式 3 のシステムの説明図を示す。この図は、ユーザアドレス空間から操作要求をダイレクトメモリアクセス (DMA) するシステムの一例を示す。従来方式 1 におけるシステムコールのオーバーヘッドを排除又は小さくするためには、ユーザアドレス空間に I/O 装置に関する制御情報をマップする必要がある。従来方式 3 においても、この点では従来方式 2 と同じである。ただし、単に I/O 装置の制御レジスタをユーザアドレス空間にマップしたのでは、複数のユーザアプリケーションからアクセスされた場合に、混乱が生じてしまう。これを解決するためには、I/O 装置に対する要求内容をメモリ上にまとめておいて、I/O 装置側に要求内容を読み出して制御レジスタにセットしてもらうようにすればよい。つまり、I/O 装置側が DMA によって制御レジスタの内容を更新する。そして、一つの要求内容が終る（区切りがつく）までは、次の要求内容の処理を開始しないように I/O 装置が動作すれば動作の混乱も起こらない。また、汎用 OS の下でメモリ資源は保護されているので、アプリケーション間で要求内容を書き潰したりする心配はない。

【0006】

ここで重要になるのは、どうやって I/O 装置に要求内容が書かれているメモリの場所を指示するかである。そこで、従来方式 3 では、要求内容を示すメモリの場所を示すポインタを記憶したコンテキストレジスタ (context register) を I/O 装置内に持たせる。コンテキストレジスタは、例えば、どのプロセス・タスクがホストプロセッサで実行されているかを示すレジスタである。OS は、どの

ユーザのタスク（プロセス）が実行しているかを把握している。そして、OSは、実行されているプロセスに応じて、コンテキストレジスタを書き換えることができる。プロセッサにあるアプリケーション（例えば、プロセスA）が割り当てられる場合には、該当アプリケーションの要求内容の格納場所をコンテキストレジスタにセットする。

【0007】

また、各アプリケーションによるユーザアドレス空間には、要求内容を登録したことを知らせるペンディングレジスタ（ペンディングレジスタ）をマップする。ペンディングレジスタは、例えば、いずれかのプロセスがI/O装置をアクセスしたことを示すフラグである。プロセスA用ページ（page for A）には、要求（requests）が記憶される。要求の内容は、例えば、I/O指令であり、I/O装置がプリンタであれば、印刷指令、改ページ指令等であり、入出力装置であれば、どの装置に出力又は入力するかの指令等である。また、従来方式3では、I/O装置はペンディングレジスタがアクセスされると、コンテキストレジスタの（ポインタの）示すメモリ位置から該当する要求内容を取り出して（DMAして）、実行する。

【0008】

【発明が解決しようとする課題】

しかしながら、従来方式1は、ユーザアプリケーションが入出力操作をする必要がある毎に、システムコールを含む一連のカーネルプログラムを呼び出す必要があり、オーバヘッドが大きくなるという課題がある。

【0009】

このため、従来方式2が提案されたものの、この従来方式2では、同時に複数のアプリケーションが該当I/O装置を使うことができない。この例では、プロセスAがI/O装置を使用している場合に、プロセスBがI/O装置を使用することができない。また、一つのアプリケーションが自分の都合だけでI/O装置の制御レジスタの内容を変更してしまうので、複数のアプリケーションが時分割で制御レジスタを操作すると、制御レジスタの操作の一貫性が崩れてしまう。よって、この従来方式2ではI/O資源を排他的に一つのアプリケーションに割り

当てて使用する必要がある。

【0010】

そこで、従来方式1のシステムコールのオーバヘッドの課題、及び、従来方式2の割り当ての課題を解決する方式として従来方式3が提案された。しかしながら、従来方式3において、I/O装置内のコンテキストレジスタは、プロセッサコンテキストの一部となり、OSがコンテキスト切替を行う毎に、プロセッサに割り当てられたアプリケーションの要求内容格納場所に更新する必要がある。コンテキスト切替は、OSの根幹に関わる動作の一つであるため、既存のOSに対してプロセッサコンテキストを拡張する改造を施すことは難しい。ましてや、デバイスドライバの追加等では、プロセッサコンテキストは拡張できない。また、I/O装置が接続される計算機がマルチプロセッサ構成である場合にはコンテキストレジスタはプロセッサ台数と同じだけ必要であり、ペンディングレジスタへのアクセスがどのプロセッサからのアクセスであるか識別可能である必要がある。I/O装置の制御回路がLSI化される場合には、マルチプロセッサ対応を考慮すると、多くのコンテキストレジスタを用意しておく必要があり、通常の単体プロセッサもしくは少数台の並列システムでは無駄なコストになってしまう。

【0011】

このように、従来行われているユーザレベルによるI/O装置へのアクセス方式はそれぞれ課題を抱えている。また、例えば、上述のように、通信およびI/O装置へのユーザレベルのアクセスを高速化するためにユーザメモリ空間に通信や入出力操作のためのレジスタをマップする手法が行われている。しかし、従来の方式では同時に複数のアプリケーションが使用することができなかった。

【0012】

本発明は、以上の点に鑑み、メモリ管理機構を流用することにより、複数のアプリケーションが同時に低コストで通信および入出力操作、インタフェース操作を可能とするアクセス方法及びアクセス処理プログラムを記録した記録媒体を提供する。

【0013】

【課題を解決するための手段】

本発明の特徴のひとつは、ページ単位のメモリエイリアス（物理的に同一の対象に複数のアドレスが割り当てられた状態）を故意に起こして、どのエイリアスアドレスからアクセスされたか検知する機構によってアクセスしたアプリケーションを同定することにある。メモリはページ単位に管理されているので、アプリケーションごとに別のエイリアスを割り当てることによって、他のエイリアスアドレスをアクセスするような不正な行動は行えない。及びアクセス処理プログラムを記録した記録媒体を提供する。

【 0 0 1 4 】

本発明の解決手段によると、

第 1 のプロセスが、入出力装置又はインタフェースのオープン処理をオペレーティングシステムに要求するステップと、

オペレーティングシステムが、第 1 のプロセスの要求格納領域を示すためのコンテキスト識別子を割り当て、該コンテキスト識別子に該当するメモリページを未処理の要求があることを示すペンディングレジスタのアクセス用アドレスとしてマップするステップと、

第 1 のプロセスが、要求格納領域へ入出力装置又はインタフェースへの要求内容を記述するステップと、

オペレーティングシステムが、ペンディングレジスタのアクセス用アドレスを使って、未処理の要求があることを入出力装置又はインタフェースに伝えるステップと、

入出力装置又はインタフェースが、ペンディングレジスタに記憶されたコンテキスト識別子に基づき、第 1 のプロセスの要求を読み出すステップを含むアクセス方法及びアクセス処理プログラムを記録した記録媒体を提供する。

【 0 0 1 5 】

【発明の実施の形態】

本発明では、上述の課題を解決する新しいユーザレベルによる I / O 装置へのアクセス方法を提供する。

【 0 0 1 6 】

従来方式 3 において、入出力処理されていない要求がメモリ上にあることを示すペンディングレジスタへのアクセスが、どのアプリケーションによってなされたかを、I/O 装置側が識別できれば、要求内容が格納されたメモリアドレスを求める方法が構成できる。極論すれば、コンテキストレジスタとペンディングレジスタの対が十分な数だけあり、アプリケーション空間ごとに一つのペンディングレジスタがマップされていれば、アクセスされたペンディングレジスタに対応したコンテキストレジスタの内容を使って、要求を読み出すアドレスを決定できる。この場合においてコンテキストレジスタをレジスタとして実装する必要はなく、I/O 装置内のメモリ上の領域等で十分代用できる。アクセスされたペンディングレジスタの位置からメモリ上のコンテキストレジスタの位置を計算すればよい。コンテキストレジスタが I/O 装置内のメモリ上の領域として実装可能であるなら、さらに、ペンディングレジスタの数を一つないしは少数に抑えることが可能になると、非常に低コストかつオーバーヘッドのない I/O 装置へのユーザレベルアクセス方法となる。

【 0 0 1 7 】

本発明では、ペンディングレジスタの実体は一つだけ用意するが、そこへのアドレスのエイリアスを複数用意し、どのエイリアスからアクセスされたか判る機構を用意することにより、ペンディングレジスタにアクセスしたアプリケーション（プロセッサコンテキスト）を同定する。以下に、本発明の実施の形態について、具体例を示して説明する。

【 0 0 1 8 】

図 1 に、本発明に関連するシステム構成図の一例を示す。

図 1 は、例えば、標準的な構成のワークステーションもしくはパーソナルコンピュータを利用して、ユーザレベルにおいて低オーバーヘッドで I/O 装置をアクセスする方式を実現するためのシステムである。

【 0 0 1 9 】

このシステムは、I/O 装置 (A) 1-1 及び I/O 装置 (B) 1-2、プロセッサ 2、キャッシュ 3、システム L S I 4、メインメモリ 5、I/O バス 6 を備える。I/O 装置 (A) 1-1 及び I/O 装置 (B) 1-2 は、内部に未処理

の要求が登録されていることを示すペンディングレジスタを備える。このペンディングレジスタは、メモリ空間にマップ（メモリマップ）されていて、プロセッサからアクセスされる（詳細は、後述する。）。プロセッサ 2 は、キャッシュ 3 と協調して、OS に従い各種処理を実行する。キャッシュ 3 は、処理の高速化のために、プログラムやデータをコピーしておく高速メモリである。システム L S I 4 は、I/O 装置（A）1-1 及び I/O 装置（B）1-2、プロセッサ 2、メインメモリ 5 の相互の間で、データの送受を制御する。メインメモリ 5 は、オペレーティングシステム（OS）、アプリケーションとしての各種プロセス、物理アドレス空間等のプログラム及びデータ等を記憶する。また、I/O バス 6 は、拡張バスとして構成することができる。

【0020】

つぎに、図 2 に、本発明に係るアクセス方法の説明図を示す。この図は、ペンディングレジスタのアドレスマップについて示したものである。

【0021】

この例では、プロセス(process) A~C 210~212、物理アドレス空間(physical address space) 22、I/O 装置 1 (I/O device) が示される。I/O 装置 1 は、ペンディングレジスタ(pending register) 23、DMA エンジン(DMA engine) 24、要求実行用のレジスタ(register) 25、内蔵メモリ 26 を備える。また、プロセス A~C 用ページ 270~272 (例えば、page for A, B) 及びプロセス A~C 用、メモリページ 280~282 (page 0~2) が物理アドレス空間に用意され、各プロセスによる要求(requests) が要求格納領域 290~292 に記憶される。

【0022】

図 3 に、本発明に係るアクセス方法の OS 及びプロセスについてのフローチャートを示す。また、図 4 に、本発明に係るアクセス方法の I/O 装置についてのフローチャートを示す。以下、図 2~図 4 を参照して、本発明のアクセス方法について説明する。

【0023】

まず、図 2 及び図 3 のフローチャートに基づき、OS 及びプロセス側に関する

処理について説明する。

OSはプロセスを管理しているので、プロセスA～Cの存在をプロセスが作成された時から把握している。オープン処理によってわかることは、オープン処理をOSに要求したプロセスが、今後I/O装置1を使用しようとしていることである。まず、プロセスから見たI/O装置を使用するための処理が以下のように実行される。

【0024】

ステップS101では、あるプロセス（例えば、プロセスA210）がI/O装置1のオープン処理をOSに要求する。このとき、プロセスは自分のI/O装置1への要求を置く領域をOSに指示する。つぎに、ステップS102では、第1に、OSはオープン処理の中で、プロセスA210に対して使っていないI/O装置1用のコンテキストID（context-ID）（例えば、ID=0）を割り当て、そのIDに該当するメモリページ280（page 0）をプロセスA210用のペンディングレジスタ23アクセス用のアドレスとしてマップする。マップした論理アドレスはオープン処理の結果としてプロセスA210にOSから通知される。また、ステップS102では、第2に、OSが、オープン処理の中で、I/O装置1内の内蔵メモリ26にプロセスA210の要求格納領域290へのポインタ（物理アドレス）を記憶する。記憶するアドレスはプロセスA210に割り当てたコンテキストIDから容易に計算可能な場所にする。つまり、コンテキストIDから要求格納領域へのポインタが取り出せる表が、内蔵メモリ上に形成される。

【0025】

ステップS103では、プロセスA210は自分の要求格納領域290へI/O装置1への要求内容を記述する。ステップS104では、要求内容の記述後、OSは、オープン処理で割り当てられたペンディングレジスタ23用のアドレスを使って、未処理の要求があることをI/O装置に伝える。具体的にはペンディングレジスタ23のアドレスに対してアクセスする。ステップS105では、I/O装置が要求格納領域290から要求を読み出す。ステップS106により、プロセスA210がI/O装置1に対して再び要求がある場合は、ステップS1

03及びS104を繰り返す。ステップS107では、プロセスAがI/O装置1のクローズ処理をOSに要求した場合、又はプロセスA210が終了した場合等には、OSはプロセスA210に割り当てたアドレスを回収すると共に、I/O装置用のコンテキストIDの回収（空きIDとして記憶）及び内蔵メモリ26上のプロセスA210の要求格納領域290へのポインタのエントリをクリアする。

【0026】

つぎに、図2及び図4のフローチャートに基づき、I/O装置側に関する処理について説明する。I/O装置側から見ると、以下のように処理が実行される。

【0027】

ステップS201では、ペンディングレジスタ23にアクセスがあったことが、先入れ先出しメモリ（FIFO）出力からわかる。ステップS202では、FIFO出力にはコンテキストIDが含まれており、内蔵メモリ26上の表を参照することにより、ペンディングレジスタをアクセスしたプロセスA210の要求格納領域290の物理アドレスがわかる。ステップS203では、DNA エンジン24により要求格納領域290の内容を読み出して、内容に沿って内部レジスタ25を制御して、要求を実現する。ステップS204により、例えば、電源断まで、この動作をI/O装置は繰り返す。

【0028】

プロセスBもプロセスCもまったく同様にしてI/O装置1を使用する。ここで、ステップS103、S104においてOSを一切介していないことがオーバーヘッド低減の観点で重要である。たとえ、ホストプロセッサで動いているプロセスが時分割で切り替わっても、コンテキストIDを含むペンディングレジスタへのアドレスを使うことによって、一意に要求しているプロセスがわかるので、プロセス切替え時にOSは一切付加的な動作が必要ないことになる。なお、従来方式3ではコンテキストレジスタを切替える必要があった。

【0029】

つぎに、図5に、ペンディングレジスタに記憶する機構の説明図を示す。

図5の一番上のメモリ構造は、I/O装置を指定する物理アドレス51を示す

。ここでは、一例として、36bit構成で0～11ビットにページオフセット(Page Offset)としてのファンクションセレクト (Function Select)、12～19ビットにプロセスのためのコンテキストID(Context-ID)、20～35ビットにファンクションセレクト (Function Select)の各領域を含む。また、図のように、ページオフセット（大抵のプロセッサでは12bitか13bit）より上位の部分に、コンテキストIDを示すフィールドを用意する（図では8bit）。ただし、コンテキストIDのフィールドは、アドレスデコードに使用されず、ペンディングレジスタのアドレスデコードに使用され得るのはコンテキストIDよりも上位のフィールドとページオフセットの部分である。ファンクションセレクト領域は、ペンディングレジスタの位置を示す。

【0030】

ファンクションセレクトは上位アドレスに関しては必ずデコードする（一意に決める）必要があり、一方、下位アドレス（オフセット部分）は、必ずしもデコードされなくても良い。オフセット部分をデコード対象とならないようにした場合、オフセット部分がどんなパターンになっていても上位アドレスのみによりペンディングレジスタにアクセスすることになる。なお、ページ単位でしかプロセスにアドレスを割り当てることはできないので、オフセット部のデコードを省略しても他のプロセスには影響がでない。逆に省略しないと、ペンディングレジスタ以外のレジスタやメモリをペンディングレジスタと同一のページ内に割り当てることができる。この場合、ページオフセット部分のビットパターンが違うアドレスが使われる。

【0031】

また、コンテキストID領域は、I/O装置を使用するプロセス（例えば、プロセスA～Cのいずれか）を示す。コンテキストID領域が8ビットであれば、同時に256のプロセスがI/O装置の使用を要求（オープン）していても対応することができる。

【0032】

ペンディングレジスタ用のアドレスデコーダ52は、物理アドレス51のファ

ンクションセレクト領域にペンディングレジスタの場所を表す固有アドレスが記憶されている場合、ペンディングレジスタを選択するとともに、物理アドレス 5 1 のコンテキスト ID をペンディングレジスタ 5 3 に記憶する。また、必要に応じて、ペンディングレジスタ 5 3 に、プロセス識別子に対応してデータ 5 4 を記憶することができる。データ 5 4 は、例えば、プロセッサ、メインメモリ、インタフェース等から与えられることができる（例えば、ストア・スワップアクセス時）。ペンディングレジスタ 5 3 の出力 5 5 には、コンテキスト ID が含まれ、場合によってはデータもそれに対応して含まれる。

【0 0 3 3】

各アプリケーションは、特定のコンテキスト ID を割り当てられ、その ID に従ったページのみをマップする。つまり、アプリケーションが異なれば、コンテキスト ID の部分が異なったアドレスで同一のペンディングレジスタへアクセスすることになる。I/O 装置側にアクセスされるたびにコンテキスト ID 部分を保持して解釈する能力があれば、どのアプリケーションがペンディングレジスタへアクセスしたか明らかにすることができる。ペンディングレジスタへのアクセスは、ユーザアプリケーションによるアクセスであり、実行されるタイミングに制約がない。I/O 装置の動作とペンディングレジスタによる新規要求の受け付けは同期されないと入出力処理が行えなくなる。このため、ペンディングレジスタ 5 3 へのアクセスをそれを構成する FIFO にバッファリングしてやる必要がある。このときにアクセス内容（書き込みなら「書き込み」であるという情報と「書き込みデータ」）のみではなく、コンテキスト ID も同時にバッファリングする。コンテキスト ID から適当な処理によってアプリケーションの要求内容を格納しているアドレス空間とアドレスを求めて、I/O 装置が要求内容を読み出し、入出力処理を行う。

【0 0 3 4】

FIFO バッファの溢れに関しては、以下のように、返り値もしくは割り込みで対処することができる。

・返り値で対処する場合は、「書き込みデータ」が不要であれば、l ad（メモリ読み出し）命令によってペンディングレジスタをアクセスし、FIFO バッファが溢

れている場合には返り値としてエラーコード（例えば-1）を返す。ユーザアプリケーションは、返り値を見て、成功するまでアクセスを繰り返す。「書き込みデータ」が必要であれば、swap（不可分読み書き）命令によって返り値を受け取りつつ、データをI/O装置1に受け渡す。

・I/O装置用の拡張バスを介したload命令やswap命令は、store命令と比べるとプロセッサにとって実行時間が長くコストが高い。そこで、割り込みで対処する場合は、store命令によってペンディングレジスタへアクセス可能にするためには、FIFOバッファ溢れを割り込みによって検出する。FIFOバッファの容量が十分にあり、I/O装置1の能力が飽和していなければ、FIFOバッファ溢れは発生しない。FIFOバッファ溢れが発生した場合には、処理コストが少しぐらい掛かってもやむを得ないと考えて、割り込みを発生させてペンディングレジスタへアクセスに失敗したアプリケーションやアクセス内容をOSによって保管しておき、I/O装置に登録可能になった時点で処理する。

【0035】

なお、本発明に係るアクセス方法は、I/O装置に限らず、適宜のインタフェースのアクセスに適用することができる。また、本発明に係るアクセス方法は、アクセス処理プログラムとして、CD-ROM等の記録媒体又はインターネット等の伝送媒体により提供されることができる。

【0036】

【発明の効果】

本発明によると、以上のように、メモリ管理機構を流用することにより、複数のアプリケーションが同時に低コストで通信および入出力操作、インタフェース操作を可能とするアクセス方法及びアクセス処理プログラムを記録した記録媒体を提供することができる。

【図面の簡単な説明】

【図1】

本発明に関連するシステム構成図。

【図2】

本発明に係るアクセス方法の説明図。

【図 3】

本発明に係るアクセス方法の OS 及びプロセスについてのフローチャート。

【図 4】

本発明に係るアクセス方法の I/O 装置についてのフローチャート。

【図 5】

ペンディングレジスタに記憶する機構の説明図。

【図 6】

従来方式 1 のシステムの説明図。

【図 7】

従来方式 2 のシステムの説明図。

【図 8】

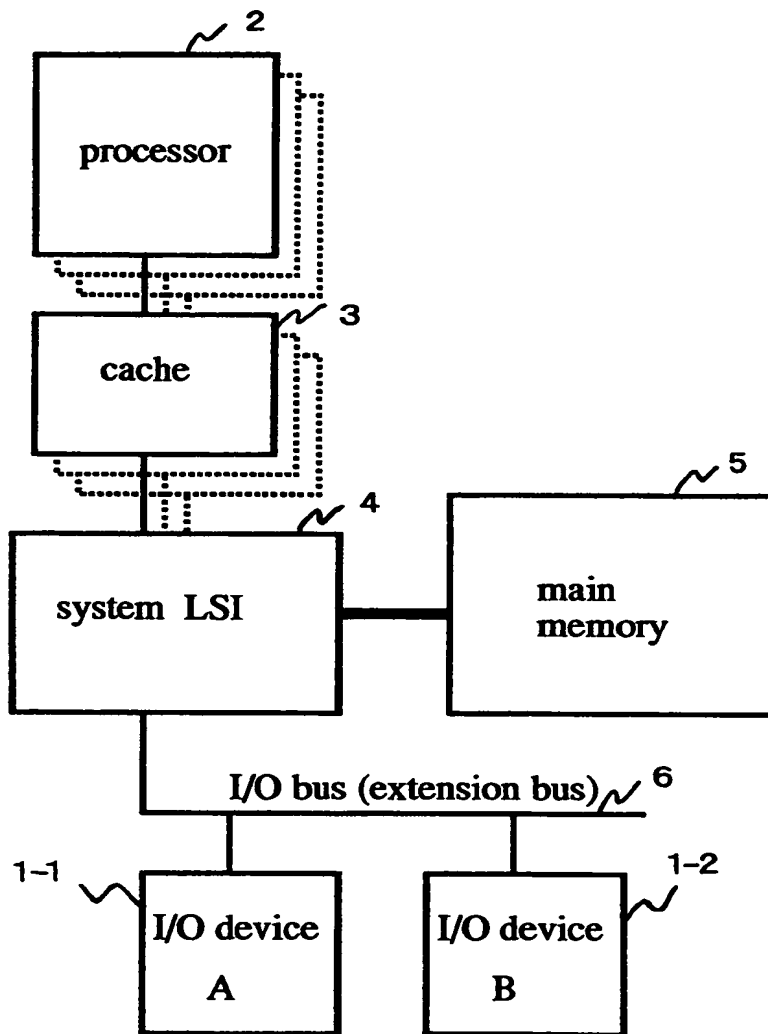
従来方式 3 のシステムの説明図。

【符号の説明】

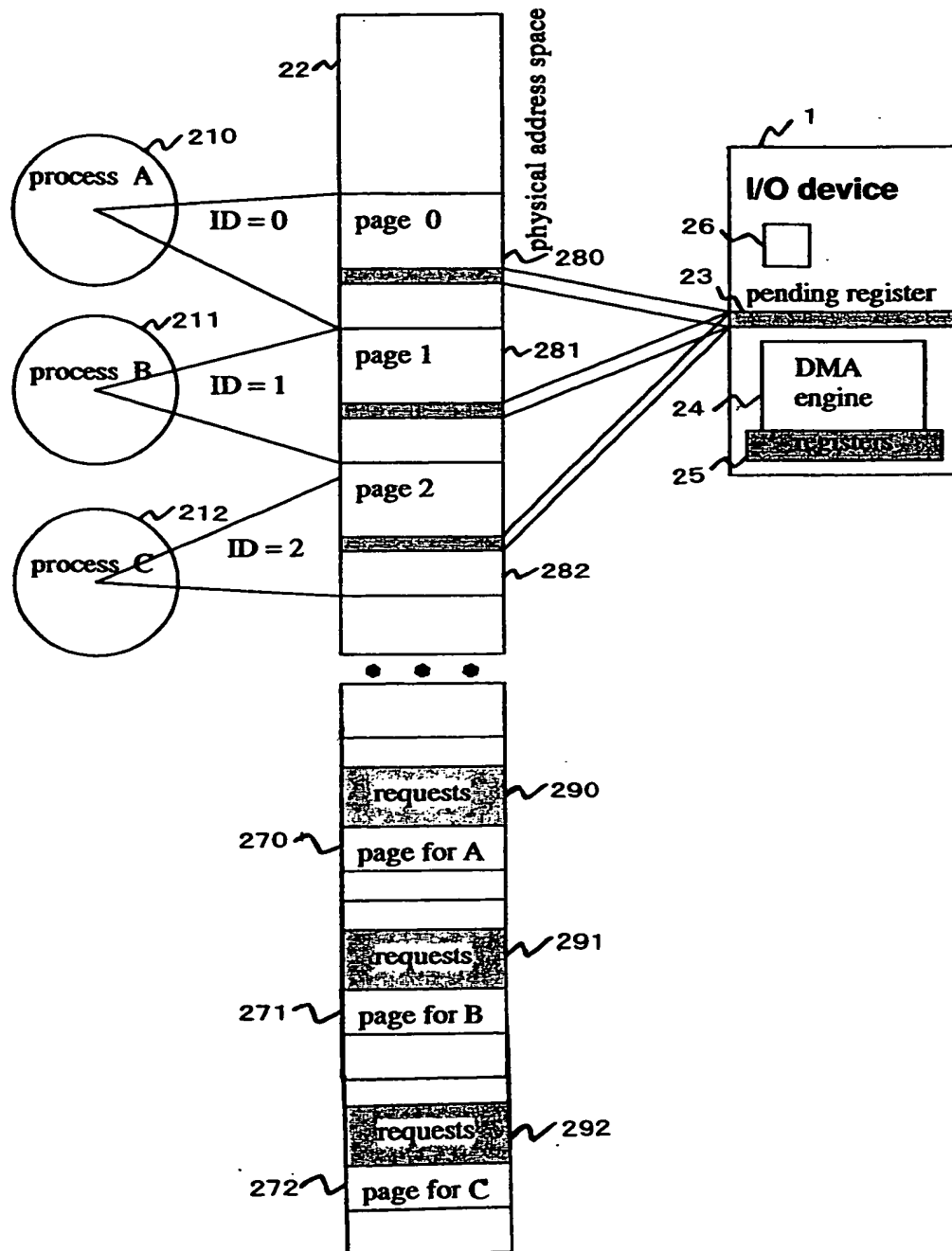
- 1 I/O 装置
- 2 プロセッサ
- 3 キャッシュ
- 4 システム L S I
- 5 メインメモリ 5
- 6 I/O バス

【書類名】 図面

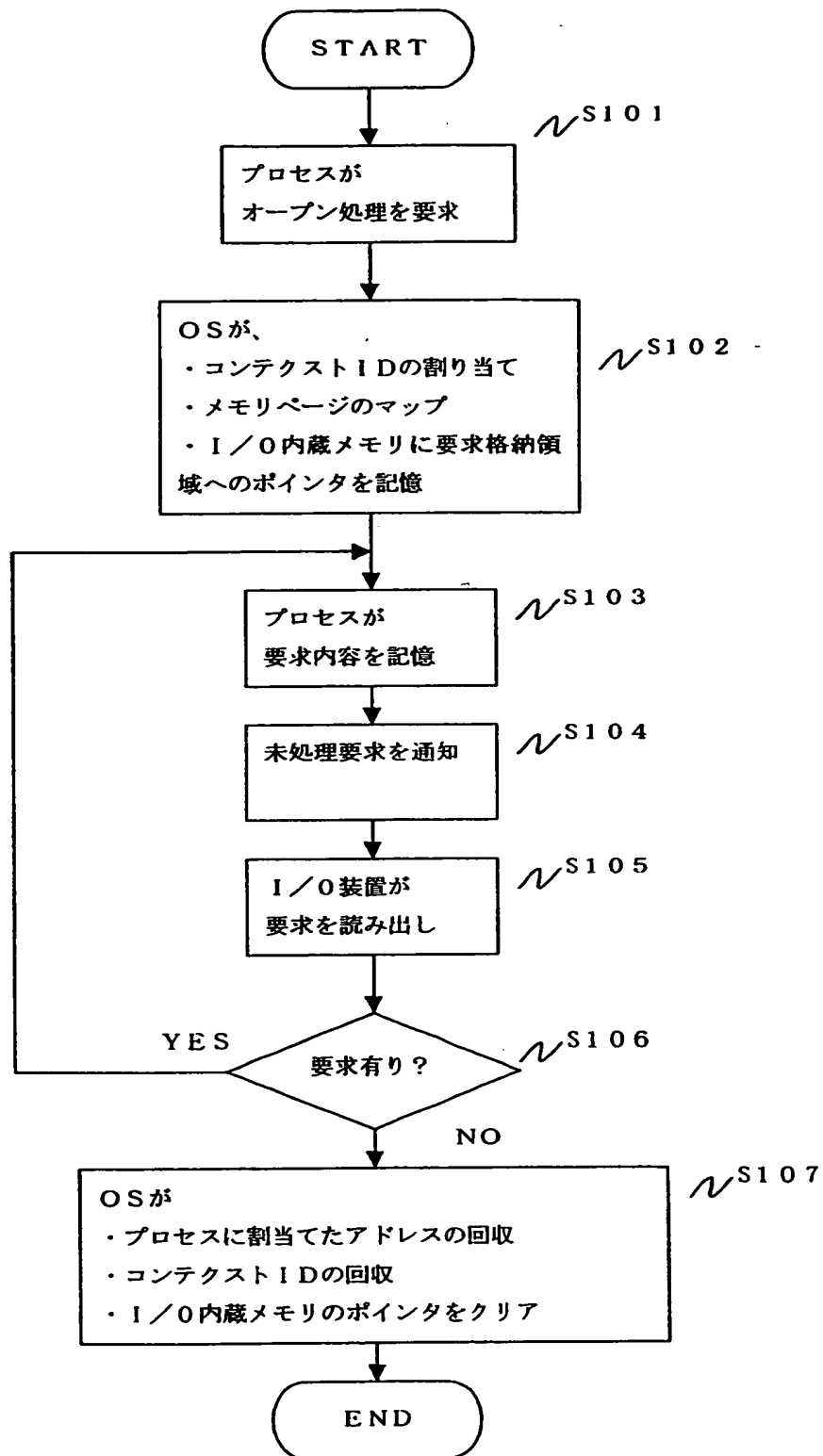
【図 1】



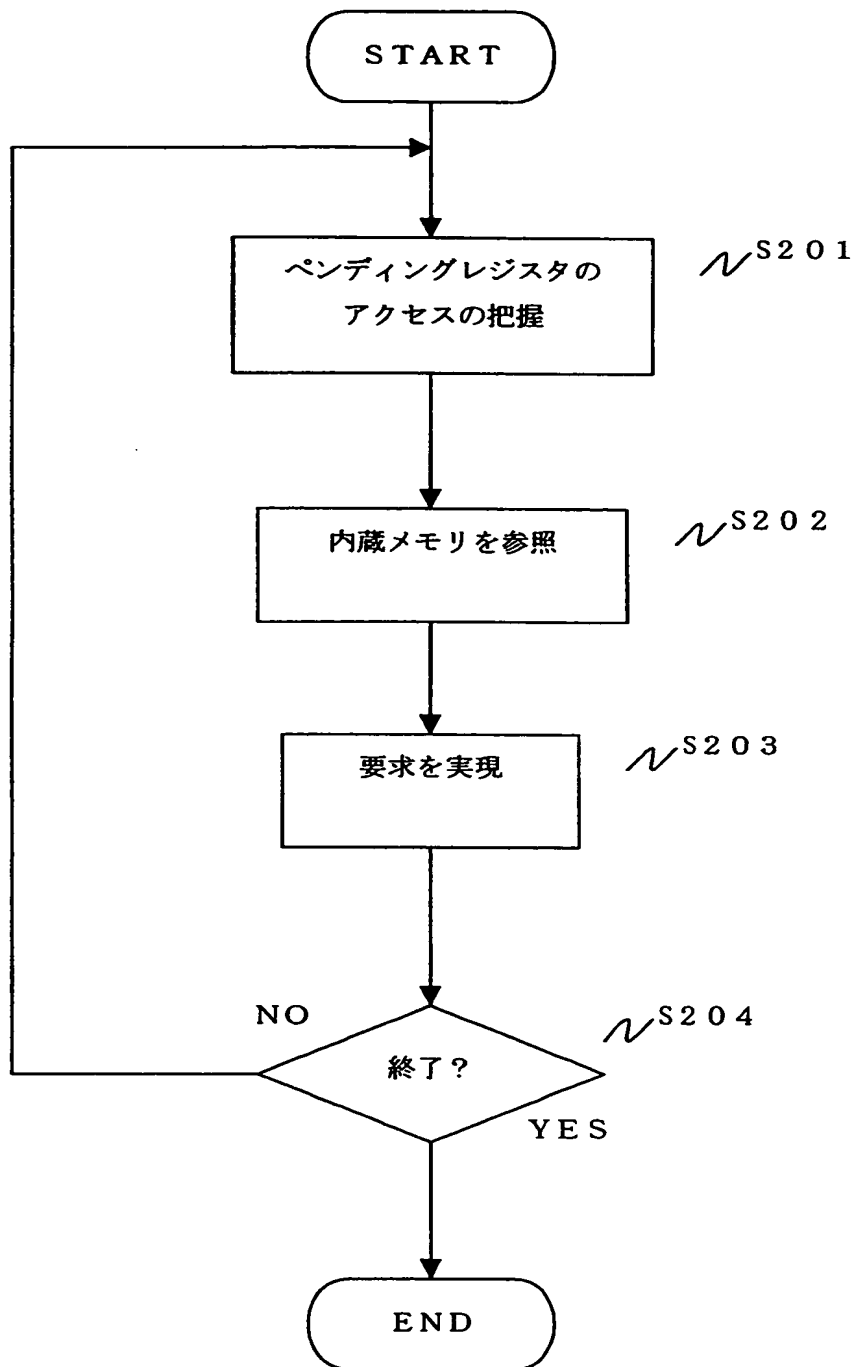
【図 2】



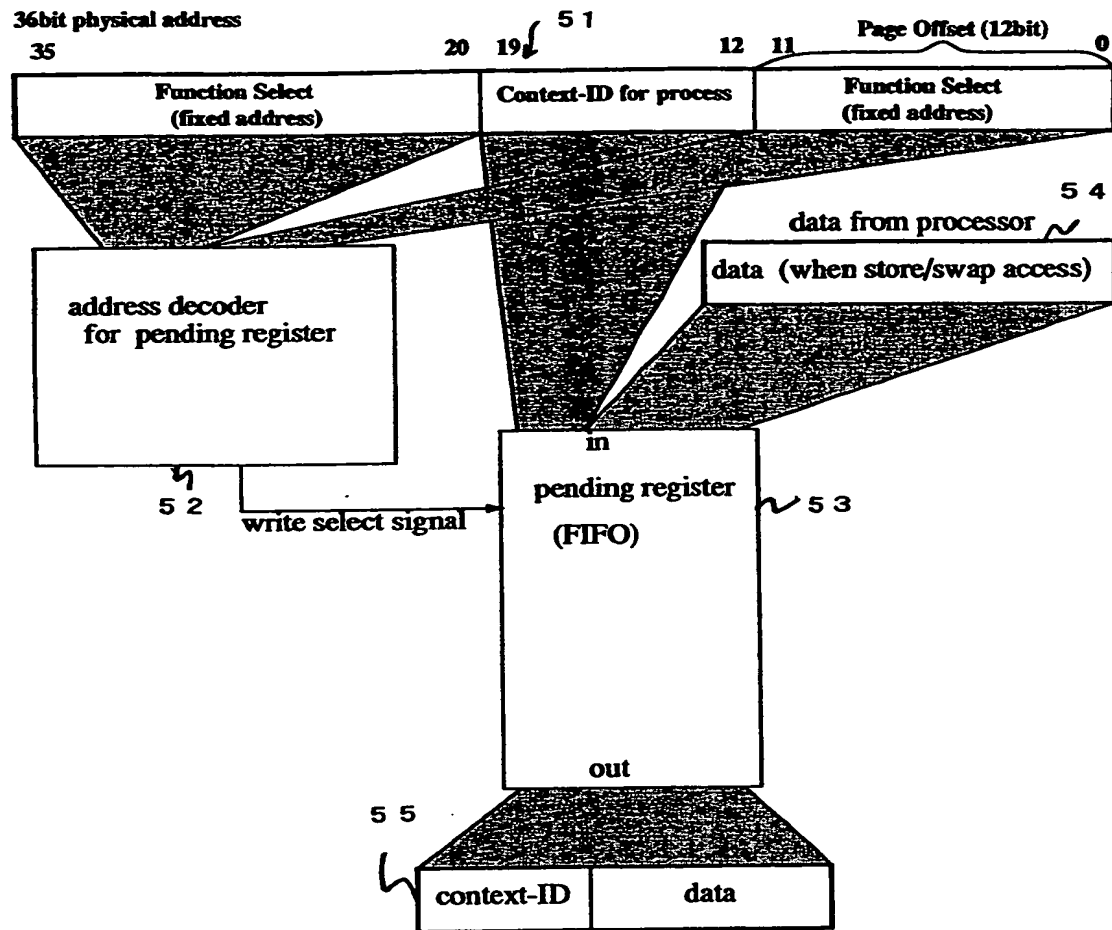
【図 3】



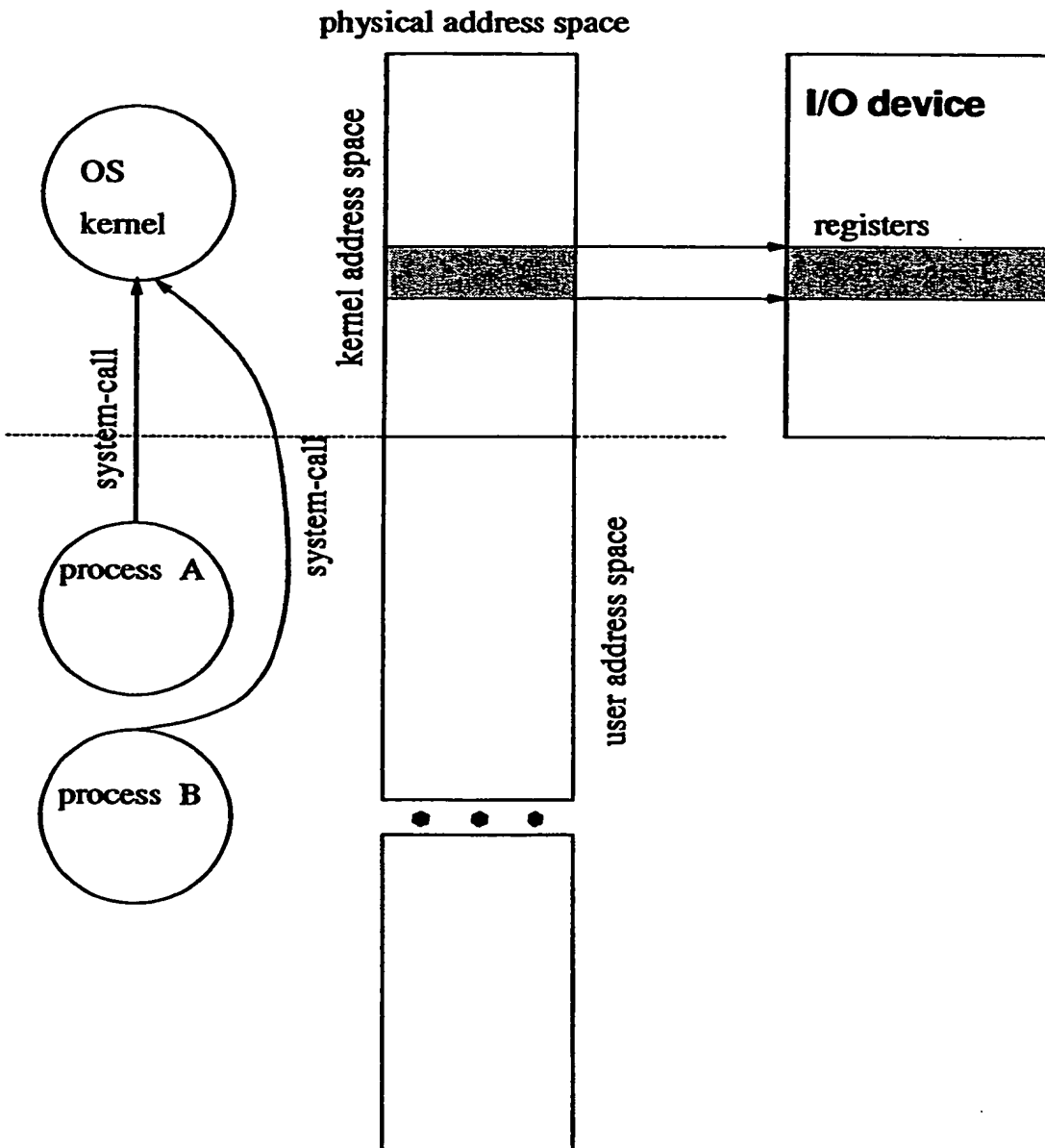
【図 4】



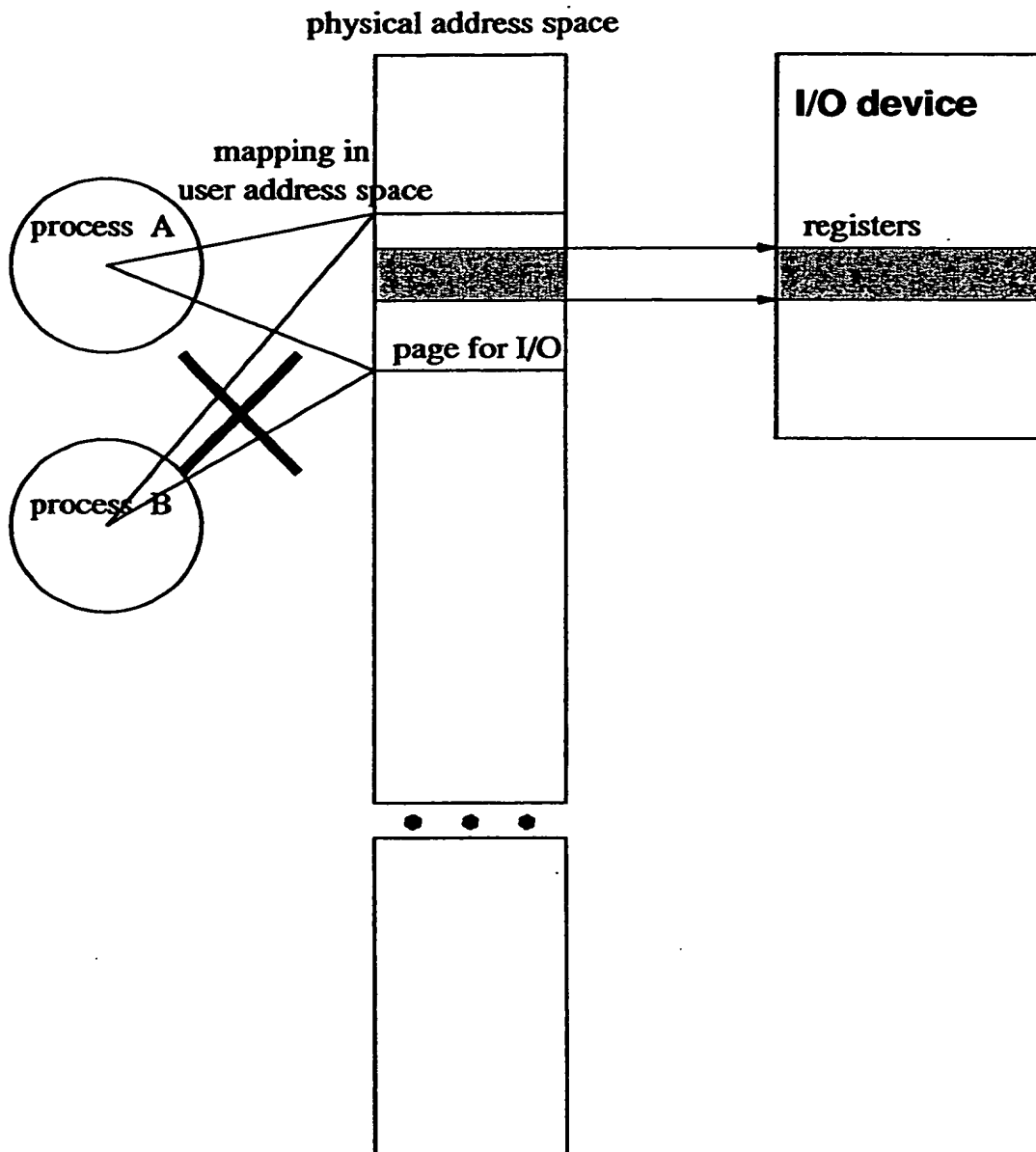
【図 5】



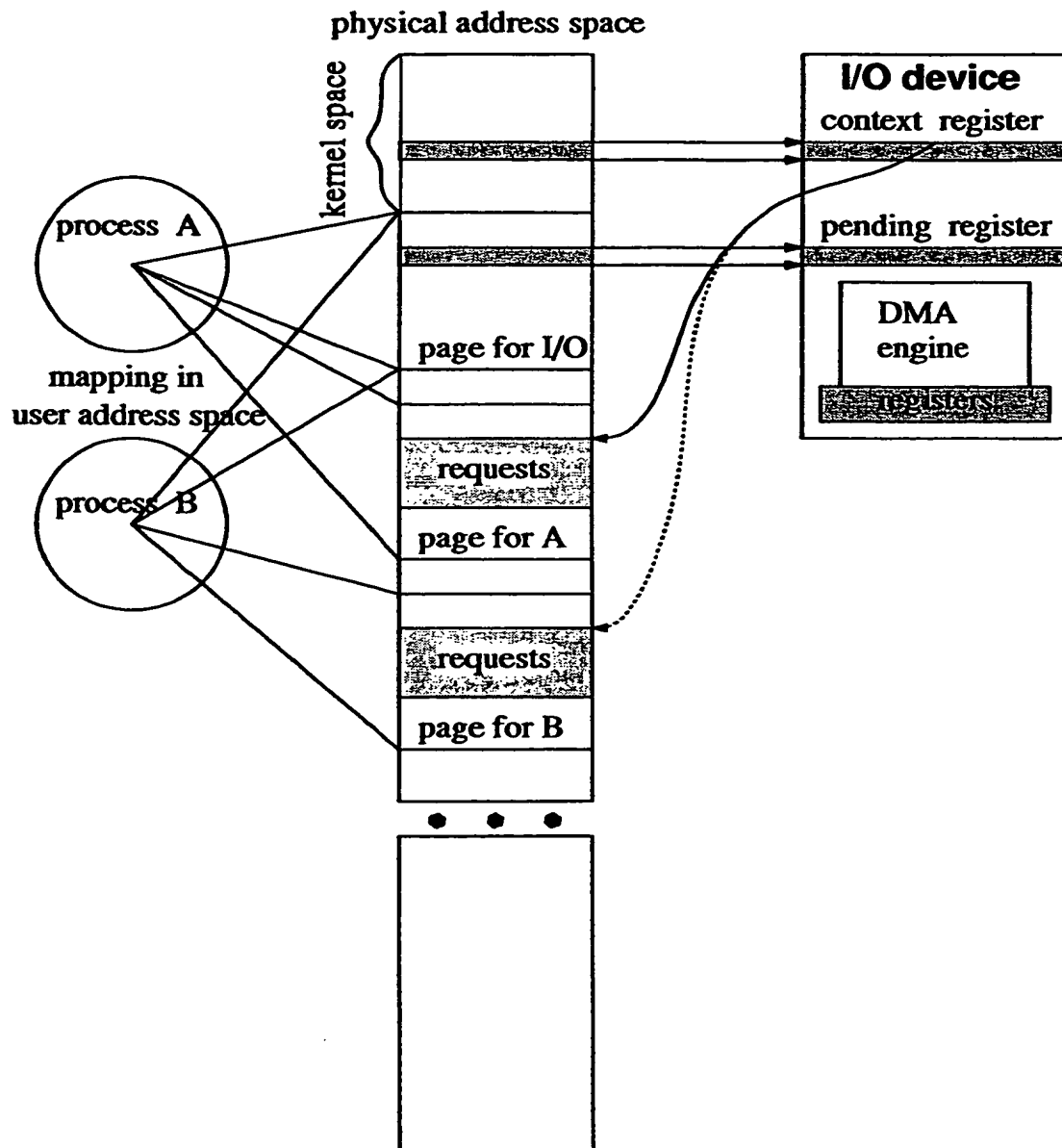
【図 6】



【図 7】



【図 8】



【書類名】 要約書

【要約】

【課題】 複数のアプリケーションが同時に低コストで通信および入出力操作、インタフェース操作を可能とする。

【解決手段】 プロセス A が I / O 装置への要求を置く領域を OS に指示する。OS は、プロセス A に対して使っていない I / O 装置用のコンテキスト ID を割り当て、その ID に該当するメモリページをプロセス A 用のペンディングレジスタアクセス用のアドレスとしてマップし、I / O 装置内の内蔵メモリにプロセス A の要求格納領域へのポインタ（物理アドレス）を記憶する。プロセス A は自分の要求格納領域へ要求内容を記述し、OS は、ペンディングレジスタ用のアドレスを使って、未処理の要求があることを I / O 装置に伝える。I / O 装置は、DMA エンジンにより要求格納領域の内容を読み出して、要求を実現する。

【選択図】 図 2

出 願 人 履 歴 情 報

識別番号 [396020800]

1. 変更年月日	1998年 2月24日
[変更理由]	名称変更
住 所	埼玉県川口市本町4丁目1番8号
氏 名	科学技術振興事業団